

Use environment variables

Incorporating environment variables into your Handler Functions is a key aspect of managing external resources like S3 buckets.

This section focuses on how to use environment variables to facilitate the uploading of images to an S3 bucket using RunPod Handler Functions.

This will go through the process of writing Python code for the uploading and setting the necessary environment variables in the Web interface.

Prerequisites

- Ensure the RunPod Python library is installed: `pip install runpod`.
- Have an image file named `image.png` in the Docker container's working directory.

Python Code for S3 Uploads

Let's break down the steps to upload an image to an S3 bucket using Python:

1. **Handler Function for S3 Upload:** Here's an example of a handler function that uploads `image.png` to an S3 bucket and returns the image URL:

```
from runpod.serverless.utils import rp_upload
import runpod

def handler(job):
    image_url = rp_upload.upload_image(job["id"], "./image.png")
    return [image_url]

runpod.serverless.start({"handler": handler})
```

[Ask AI](#)

2. **Packaging Your Code:** Follow the guidelines in [Worker Image Creation](#) for packaging and deployment.

Setting Environment Variables for S3

Using environment variables securely passes the necessary credentials and configurations to your serverless function:

1. **Accessing Environment Variables Setting:** In the template creation/editing interface of your pod, navigate to the bottom section where you can set environment variables.
2. **Configuring S3 Variables:** Set the following key variables for your S3 bucket:
 - `BUCKET_ENDPOINT_URL`
 - `BUCKET_ACCESS_KEY_ID`
 - `BUCKET_SECRET_ACCESS_KEY` Ensure that your `BUCKET_ENDPOINT_URL` includes the bucket name (e.g., `https://your-bucket-name.nyc3.digitaloceanspaces.com`, `https://your-bucket-name.nyc3.digitaloceanspaces.com`).

Testing Your API

Finally, let's test the serverless function to confirm that it successfully uploads images to your S3 bucket:

1. **Making a Request:** Make a POST request to your API endpoint with the necessary headers and input data. Remember, the input must be a JSON item:

```
import requests

endpoint = "https://api.runpod.ai/v2/xxxxxxxxx/run"
headers = {"Content-Type": "application/json", "Authorization": "Beare"}
input_data = {"input": {"inp": "this is an example input"}}

response = requests.post(endpoint, json=input_data, headers=headers)
```

2. **Checking the Output:** Make a GET request to retrieve the job status and output. Here's an example of how to do it:




Ask AI

```
response = requests.get(
    "https://api.runpod.ai/v2/xxxxxxxx/status/" + response.json()["id"]
    headers=headers,
)
response.json()
```

The response should include the URL of the uploaded image on completion:

```
{
  "delayTime": 86588,
  "executionTime": 1563,
  "id": "e3d2e250-ea81-4074-9838-1c52d006ddcf",
  "output": [
    "https://your-bucket.s3.us-west-004.backblazeb2.com/your-image.png"
  ],
  "status": "COMPLETED"
}
```

By following these steps, you can effectively use environment variables to manage S3 bucket credentials and operations within your RunPod Handler Functions. This approach ensures secure, scalable, and efficient handling of external resources in your serverless applications.

 [Edit this page](#)

Previous

« [Test locally](#)

Next

[Test response time](#) »

Docs

[Overview](#)

[Tutorials](#)


[AI APIs](#)

[Community](#)




Ask AI

Discord 

Contact us 

More

Blog 

GitHub 

Copyright © 2024 RunPod



Ask AI